It's Good to Relax: Fast Profit Approximation for Virtual Networks with Latency Constraints

Robin Münk*, Matthias Rost†, Harald Räcke*, Stefan Schmid‡
*Technical University of Munich
†SAP SE & Technische Universität Berlin
‡Faculty of Computer Science, University of Vienna

***Abstract*—This paper proposes a new approximation algorithm for the offline Virtual Network Embedding Problem (VNEP) with latency constraints. Our approximation algorithm FLEX allows for (slight) violations of the latency constraints in order to greatly lower the runtime. It relies on a reduction to the Restricted Shortest Path Problem (RSP) and leverages a classic result by Goel et al. We complement our formal analysis with a simulation study demonstrating our algorithm's computational benefits. Our results generalize to any other additive edge metric, as e.g., hop count or even packet loss probability.**

2021 IFIP Networking Conference (IFIP Networking) | 978-3-9031-7639-3/21/$31.00 ©2021.10.23919/IFIPNETWORKING52078.2021.9472197

I. Introduction

The Virtual Network Embedding Problem (VNEP) is a fundamental resource allocation problem in networks and has received significant interest in the network algorithms community over the last decade. Given are a set of request graphs (the virtual networks, sometimes also called "guest graphs") and a single substrate network (the physical infrastructure, also called the "host graph"). For every request graph the task is to either find a feasible embedding that maps each request node to a substrate node and every request edge to a path in the substrate graph, or to reject the request. The cumulative resource consumption of the embeddings may not violate the substrate capacities on both nodes and edges. In this paper we additionally consider latency constraints, restricting the length of request edge embeddings. Every admitted and feasibly embedded request yields a given profit and the goal is to maximize the total profit.

The VNEP is hard so solve in many variants. Even when neglecting the cumulative feasibility constraints, known as the Valid Mapping Problem (VMP) [3], it remains $\mathcal{NP}$-hard [5]. Importantly, solving the VMP is an essential building block for approximating the VNEP [3].

This paper presents a novel, fast and practical approximation algorithm FLEX for the VNEP with latency constraints. FLEX provides both analytical approximation guarantees and performs well in practice, as demonstrated in our computational evaluation. FLEX is based on the insight that a slight relaxation of the latency guarantees can result in significantly faster and hence more practical solutions. The latency violations can be made arbitrarily small, by trading off for a longer runtime.

Annex to ISBN 978-3-903176-39-3

To achieve this, FLEX builds upon the dynamic programming and randomized rounding framework by Rost et al. [3], which solves an all-pairs Restricted Shortest Path Problem (RSP) as a subroutine. In order to solve the RSP, we employ a classic result by Goel et al. [1] which allows, in one execution, to calculate the routes for all destination nodes at once for a given source node.

Compared to the state-of-the-art algorithm, referred to as STRICT, which provides strict latency guarantees, FLEX is orders of magnitudes faster, sometimes reducing the runtime from over nine hours to below three minutes. At the same time, the profit approximation and average latency achieved by FLEX is similar to the one obtained by STRICT, making FLEX a much more practical solution. A full account of the algorithm together with an extended evaluation is given in our technical report [2].

II. Model and Preliminaries

The **substrate network** is given as a directed graph $G_S = (V_S, E_S)$. Each component of the network, i.e., each substrate node $v_S \in V_S$ and each substrate edge $e_S \in E_S$, has a capacity $d_S : G_S \to \mathbb{R}_{\geq 0}$. For nodes, the capacity may refer, e.g., to the number of available CPU cores, and restricts the number of virtual nodes that can be mapped onto it. Further, each substrate component $x \in G_S$ may be attributed with a cost value $c_S(x) \in \mathbb{R}_{\geq 0}$ for its usage. Substrate edge latencies are given by $l_S : E_S \to \mathbb{R}_{\geq 0}$ and represent the time delay between adjacent nodes.

A **request** is likewise represented by a directed graph $G_r = (V_r, E_r)$ with demands $d_r : G_r \to \mathbb{R}_{\geq 0}$ for each virtual component and an associated latency bound $T_r \in \mathbb{R}_{\geq 0}$ such that all virtual edges of E_r must be embedded with a lesser or equal latency. Every request r yields a given profit $b_r \in \mathbb{R}_{\geq 0}$ if it is embedded.

A **mapping** represents how a request is embedded in the substrate. We allow the specification of a set of forbidden nodes and edges with each request. Formally, a *valid* mapping of request r onto the substrate G_S is defined as a tuple $m_r = (m_r^V, m_r^E)$ of functions, such that:

- The function $m_r^V : V_r \to V_S$ assigns a *valid* substrate node to every virtual node. A substrate node is valid if it is not forbidden and has sufficient capacity.
- The function $m_r^E : E_r \to \mathcal{P}_S$ maps each virtual edge $(i, j) \in E_r$ to a *valid* simple path in the substrate network connecting $m_r^V(i)$ to $m_r^V(j)$.

With regard to latencies, a mapping is further called *valid*

- *under the strict latency constraint* if it additionally fulfills $\sum_{(u,v)\in m_r^E(i,j)} l_S(u,v) \leq T_r$ for all $(i,j) \in E_r$ such that all latency bounds are met exactly, or
- *under a* $(1+\epsilon)$*-approximate latency constraint* for some $\epsilon > 0$ if it is valid and fulfills

$$\sum_{(u,v)\in m_r^E(i,j)} l_S(u,v) \leq (1+\epsilon) \cdot T_r$$

for $(i,j) \in E_r$, allowing for small latency violations.

For a valid mapping $m_r = (m_r^V, m_r^E)$ and substrate element $x \in G_S$ the induced resource allocation $A(m_r, v)$ is the sum of the demands of all request nodes or edges that are mapped onto x by m_r. For a single request r the **Valid Mapping Problem** (VMP) asks to find a valid mapping m_r that minimizes the mapping cost $c(m_r) = \sum_{x\in G_S} c_S(x) \cdot A(m_r, x)$.

For the definition of the VNEP a set of requests $\mathcal{R}$ is given. We refer to a set of mappings $\{m_r\}_{r\in\mathcal{R}'}$ for a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ as a feasible embedding iff. the cumulative resource allocation on any substrate element does not exceed its capacity, i.e., if for all $x \in G_S$ it holds $\sum_{r\in\mathcal{R}'} A(m_r, x) \leq d_S(x)$. It is important to note that the validity of mappings only considers the feasibility of single node and edge mappings while the feasibility of embedding takes the cumulative resource allocations of a *set* of mappings into account. The (offline) **Virtual Network Embedding Problem** (VNEP) then is to find a feasible embedding $\{m_r\}_{r\in\mathcal{R}'}$ of a subset of given requests $\mathcal{R}' \subseteq \mathcal{R}$ which maximizes the profit $\sum_{r\in\mathcal{R}'} b_r$.

The **Restricted Shortest Paths Problem** (RSP) is an important subproblem when solving the VNEP with latencies. For the RSP a directed graph $G = (V, E)$ is given, where each edge $e \in E$ is associated with a cost c_e and a latency l_e, both non-negative. Then for a given source $s \in V$ and target $t \in V$ the goal is to find a cost-minimal path from s to t such that the path's latency does not exceed a given limit $T \in \mathbb{R}_{\geq 0}$. For the purposes of this paper, the graph G will be the substrate G_S and the limit T will be equal a latency bound T_r of a request.

III. Algorithm and Evaluation

In order to approximate the VNEP with latency constraints, we build upon the framework by Rost et al. [3] which is parametrized by the treewidth of the request graphs, a measure of similarity to trees. As a result the algorithm's runtime is only polynomial if the maximal treewidth of the request graphs is a constant.

Their approach tackles the problem in multiple steps. First, for each request graph G_r a tree decomposition $\mathcal{T}_r$ of limited treewidth $tw(\mathcal{T}_r)$ is computed. It is then shown that the Valid Mapping Problem (VMP) can be solved on this tree representation using the DynVMP algorithm using dynamic programming (in time and space exponential in the request's treewidth) [3]. Given the ability to solve the VMP (without latencies), the *fractional* VNEP is then shown to be solvable via column generation techniques where the DynVMP algorithm is used as a separation oracle. The thus computed fractional solution can then be interpreted as a 'probability distribution' over the valid mappings constructed in the column generation step and can be easily converted into a solution to the VNEP via (repeated) randomized rounding. Altogether this approach results in an algorithm that produces approximate solutions to the VNEP *without latencies*.

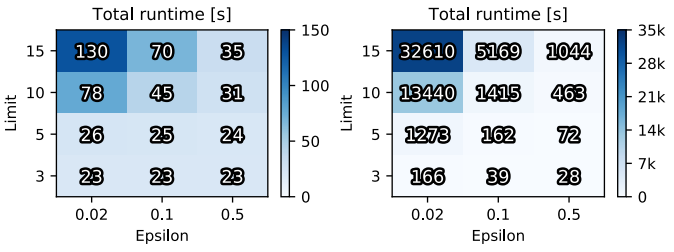
As latencies only change the notion of validity of mappings and pertain to individual request graphs, the DynVMP algorithm needs to be adapted to return solutions respecting latency constraints. This restriction is handled when edge mappings are calculated by approximating the underlying Restricted Shortest Paths problem for each pair of substrate nodes and each request edge. As the RSP needs to be solved for every pair of substrate nodes, the *All-Pairs Restricted Shortest Path Problem* (APRSP) needs to be solved. Given the $\mathcal{NP}$-hardness of the RSP, the APRSP can only be approximated.

We propose the algorithm Flex to solve the APRSP subproblem, which results from using the procedure by Goel et al. [1] to calculate latency-constrained shortest paths for the DynVMP algorithm. This approach comes with a trade-off. The algorithm by Goel et al. calculates cost-optimal paths at the expense of allowing for a violation of the latency constraint by a factor of up to $(1+\epsilon)$. The approach starts with a coarse scaling of the edge latencies to integers. The modified problem is solved exactly using dynamic programming resulting in cost-minimal paths for a weakened latency constraint. If all paths are also valid for the $(1+\epsilon)$-approximate constraint, the algorithm terminates. Otherwise the process is repeated with a finer scaling until a solution is found.

The crucial advantage of the procedure by Goel et al. are that one execution gives the results for *all* destination nodes at once for a given start node. This leads to a significant decrease in runtime as it only has to be executed $|V_S|$ times to produce paths between all pairs of source and target nodes. This subroutine, APRSP_Goel, only requires $|V_S|$ calls to Goel et al.'s algorithm to prepare the cost and path tables for DynVMP.

Besides the fewer required subroutine calls, the algorithm has some additional benefit. Specifically, the algorithm progressively improves the approximation's quality which allows for early stopping of the algorithm in the case of strict computation time limits. In that case, if a path has been computed, it is cost-optimal for some weaker latency constraint.

Theorem 1 (Flex). *For $n \geq 3$ the* Flex *algorithm finds a solution to the VNEP under $(1+\epsilon)$-approximate latency constraints with a profit of at least $1/3$ of the optimal profit and resource augmenations as in [3] with high probability.*



(a) FLEX (b) STRICT

Fig. 1: Total runtime for different values of limit and ϵ, split by algorithm type. The results are averaged for ERF, NRF and over all topologies.

The runtime is bounded by $\mathcal{O}(poly(\tau_{Flex}))$ *with*

$$\tau_{Flex} = \sum_{r \in \mathcal{R}} n^2 \cdot \left(|V_r|^3 \cdot n^{2 \cdot tw(\mathcal{T}_r)} + \frac{m + n \log n}{\epsilon} \right).$$

Proof. The proof is given in [2]. □

To complement our theoretical contribution we implemented FLEX and the current state-of-the-art algorithm STRICT as proposed by Rost et al. [3][1]. STRICT approximates the RSP while strictly enforcing latency limits [3].

Our first evaluation considers five real-world networks from the Topology Zoo with the number of edges ranging from 20 to 62. To impose meaningful latency limits on substrates of different sizes, we set the limit per request via scaling the average edge latency, computed based on the geographic information of adjacent nodes. The general experiment design closely follows Rost et al. [3], [4], using node and edge resource factors (NRF and ERF) to control resource scarcity. To enforce the distributed placement of nodes, each virtual node may only be mapped to a quarter of the substrate nodes.

Figure 1 depicts the runtime of the respective algorithms as a function of the approximation guarantee ϵ and the latency limit scaling factor. Clearly, FLEX provides a much better scalability both in terms of the latency limit and the approximation guarantee: for the highest latency limit and the smallest approximation factor STRICT's runtime averages to about 9 hours while FLEX takes less than 2.2 minutes. Importantly, In Figure 1a we can observe another favorable quality of the FLEX algorithm. Specifically, for small limit values (3 and 5) when very few mappings are generated, FLEX shows no change in runtime for different values of ϵ.

We observe that the profits of the FLEX and the STRICT algorithm are very similar: Figure 2 (top) shows the averaged profit across topologies for the worst approximation factor $\epsilon = 0.5$ and a medium latency limit factor of 10. Besides the profits of FLEX and STRICT generally lying close together, we note that the baseline's profit only slightly exceeds the latency limited ones.

[1]The code can be found at https://github.com/vnep-approx-latency

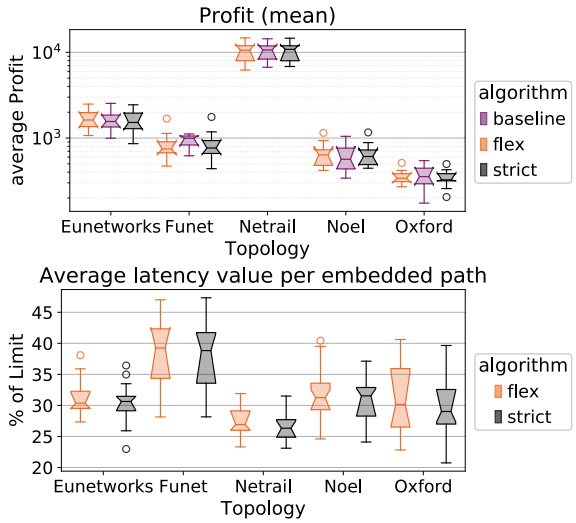


Fig. 2: Boxplot of the average achieved profits (top) and latency (bottom) by topology for $\epsilon : 0.5$, limit: 10.

Furthermore, the *average* edge latency of FLEX only slightly exceeds the one of STRICT and both strictly lie below 50% of the limit in all topologies. The experiments also showed that the latency of the FLEX algorithm rarely reaches its $(1+\epsilon)$-approximated upper bound [2].

In a separate evaluation on larger substrates (between 122 and 158 edges), on which using STRICT was computationally prohibitive, the scalability of FLEX alone was studied and it was shown that it can produce high-quality solutions within a reasonable amount of time [2].

IV. CONCLUSION

This paper presented a novel approximation algorithm for the embedding of virtual networks with latency constraints. Our algorithm is significantly faster than state-of-the-art algorithms, as we have also shown empirically.

ACKNOWLEDGMENTS.

This project received funding from the European Research Council (ERC) under grant agreement 864228 (AdjustNet), Horizon 2020, 2020-2025.

REFERENCES

[1] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. In *Proceedings IEEE INFOCOM 2001*, pages 854–858. IEEE, 2001.

[2] R. Münk, M. Rost, S. Schmid, and H. Räcke. It's Good to Relax: Fast Profit Approximation for Virtual Networks with Latency Constraints. Technical Report arXiv:2104.09249 [cs.NI], April 2021.

[3] M. Rost, E. Döhne, and S. Schmid. Parametrized complexity of virtual network embeddings: dynamic & linear programming approximations. *ACM SIGCOMM Computer Communication Review*, 49(1):3–10, 2019.

[4] M. Rost and S. Schmid. Virtual network embedding approximations: Leveraging randomized rounding. *IEEE/ACM Transactions on Networing*, 27(5):2071–2084, 2019.

[5] M. Rost and S. Schmid. On the hardness and inapproximability of virtual network embeddings. *IEEE/ACM Transactions on Networking*, 28(2):791–803, 2020.